DOSPLUS NEWS INFORMATION CENTER

September / October, 1982

$1.75

M I C R O T E R M

More and more hardware and communications services are allowing speeds up to 1200 baud. Soon, some may be going faster than that. Today's terminal software simply can't keep up. But now there is an alternative. Micro-Systems Software introduces MicroTerm, the high speed terminal.

Model III MicroTerm will communicate, without insertion of null characters, at 4800 baud. Guaranteed. No cop-outs, no question. MicroTerm is so fast that you can exit from the terminal to the main menu, adjust video width, open the buffer, turn on the printer, or any one of dozens of other functions, and return to the terminal mode **without missing a thing!**

MicroTerm continues to input from the RS232, even while at the main menu. This is the only terminal capable of such an astounding feat. MicroTerm offers you most of the features that "Brand X" smart terminals have, plus it gives you: • Ultra high baud rate operation (up to 9600 in certain cases). • Input while at menu. • Easy to use translation tables. • Easy to use phone number listings. • Maximum auto dial support — most major brands. • Direct file transfer companion program included at no exta cost (compatible with DFT). • DOS commands from menu without exiting program. • Over 34K of capture buffer (in a 48K TRS-80). • Can be set to automatically dial telephone and transmit buffer at preset time without **any** operator intervention.

And many, many more great features. MicroTerm is so fast you must see it to believe it. The various menus are displayed so fast, they seem to jump out at you. Status of various functions can be displayed and altered in split seconds.

For the computerist who wants the ultimate, state-of-the-art terminal software, there **is** no other choice.

MicroTerm retails for $79.95, but registered DOSPLUS owners can purchase it for only $59.95. $20.00 off the retail price! MicroTerm comes complete with the terminal program, the direct file transfer program, some standard translation tables, and documentation.

Don't delay, order yours today! Specify when ordering: Model I or III and whether you want it on 40 or 80 track media. Requires a 16K TRS-80 with one disk drive. We recommend 48K for serious communications work. MicroTerm will be available beginning June 30, 1982.

MICRO-SYSTEMS SOFTWARE, INC.

4301-18 Oak Circle
Boca Raton, FL 33431
Telephone: (305) 983-3390
800-327-8724

DOSPLUS NEWS INFORMATION CENTER

MICRO-SYSTEMS SOFTWARE, INC.

4301-18 Oak Circle
Boca Raton, FL 33431
Telephone: (305) 983-3390
800-327-8724

PUBLISHER
Lawrence B. Studdard

EDITOR-IN-CHIEF
Mark R. Lautenschlager

TECHNICAL EDITOR
Todd N. Tolhurst

CONTRIBUTING EDITORS
S. W. Karl
Richard Leun
Christine R. Walczak

CIRCULATION
Janet Dewey
Nancy Nickles

The DOSPLUS NEWS INFORMATION CENTER is published bimonthly by Micro-Systems Software, Inc., as a service to its customers. Micro-Systems Software, Inc. takes no responsibility or liability for typographic errors.

ADVERTISING RATES

Contact the Business Office at Micro-Systems Software, Inc. for a DNIC rate card.

(305) 391-5033

September/October

Table of Contents

MICRO-80

Since the birth of the microcomputer, users have found more and more applications for these versatile machines. Telecommunications is certainly one of the fastest growing fields in microcomputing. Large computing services offer the microcomputer user the opportunity to access vast databases of news and information from the comfort of home. Businesses use computer networks to exchange information with field representatives. Hobbiests confer through "message bases" on all sorts of computing topics.

MICRO-80 is a bulletin board program for the TRS-80 Models I and III. With MICRO-80, a TRS-80 can become a mini-communications network, allowing anyone with access to a terminal to exchange messages and information over the telephone. MICRO-80 can be public, so anyone may call and use the system, or it may be private, allowing only those persons authorized for MICRO-80 use to access the system. MICRO-80 can even be configured to provide limited access to certain users.

Users may communicate with other MICRO-80 users by leaving messages on the system. In this way, MICRO-80 acts as an electronic mail system. Users may "upload" data and program files to MICRO-80 for later "downloading" by other users.

MICRO-80 provides some of the most advanced and most convenient features ever incorporated in a microcomputer bulletin board system including:

* Message reformatting. Messages are always displayed in full-screen width, regardless of the original width of the text.

* Personal messages. Users may send private messages that may be read only by the addressee.

* Multi-level system security. MICRO-80 allows up to 15 levels of security in both the message base and the upload/download database. Each user may be individually "enabled" or "disabled" for access to each level.

* Permanent user records. MICRO-80 "remembers" each regular user and stores terminal information, security access levels, last message retrieved during last call, and other data.

* Individual user passwords. MICRO-80 users may assign themselves a "password", which prevents system access by unauthorized callers.

* File uploading/downloading. Users may "upload", or transmit, data and programs files to MICRO-80. These programs may then be "downloaded" to other MICRO-80 users.

* MICRO-80 is self-maintaining. Message space is automatically reclaimed when messages are deleted, freeing the SYSOP from tedious system maintainence.

See your local Micro-Systems Software Dealer for further information.

Log-in

Welcome to the September/October issue of the DOSPLUS NEWS INFORMATION CENTER. We certainly hope that you will find this issue worth the wait. We are sorry for the delay in producing the newsletter this time, but your editor was busy with the DOSPLUS II manual.

For those of you who are counting, this is issue number 5 of the newsletter (seems like only yesterday that we started this thing, doesn't it?). This issue, the feature article deals with data file management techniques in BASIC programming including a detailed explanation of how to write an ISAM from BASIC. These techniques have been used here to speed up our own programs and I hope that they will be able to aid you.

Todd Tolhurst has once again put together his "Random Routines" column, with a driver program for the popular TRSWATCH from the California Word Exchange contributed by Samuel J. Porter. Todd's program, "Shrink/Bas", will allow you to de-allocate space from a random disk file.

We have reviews this month and a brand new educational column called "Karl's Klassroom", written by our newest contributing editor and the first member of my staff not directly in the employ of Micro-Systems, S. W. Karl.

We do hope that you enjoy the new format. Next issue, we hope to go to a columnar text format as well. We are working hard to upgrade your newsletter. I would like to give credit for the new internal layout to our very own technical editor, Todd Tolhurst.

We are seeking software reviewers. If you have a background in business, accounting, or computing, and have some free time to write for the newsletter, contact us here at Micro-Systems. To avoid any misunderstandings, allow me to say that we are not talking about paid authors. This will be strictly those of you who enjoy writing, enjoy reviewing software, and enjoy seeing your name in print. If you are interested, please get in touch.

The DOSPLUS II project has turned out to be huge success. We would like to thank each of our Beta testers and those that aided us in any way with this project; we couldn't have done it without you.

Mark R. Lautenschlager
Editor

proper data file indexing techniques in BASIC
or
What in the world is this ISAM stuff all about?

This issue, our feature article covers the proper method of data file indexing in BASIC. This topic is one that has needed a good discussion on it for a long time, since many of the questions that I answer every day to users involve how they should be handling the various programming chores that present themselves.

As with the topic chosen last month (Input@), this one requires that I cover two distinct areas :

(1) History of data file management techniques.

(2) Current ISAM style techniques.

We will cover each of these in turn. First let me establish some ground rules. We will be working with random files as we are discussing this technique. If you do not have a good working understanding of random files, then you are cheating yourself by trying to apply the information given here. Go back and first become proficient at working random files and moving data in and out of the file buffer. THEN you are ready to consider some of the ideas presented here.

The way things used to be -

In the very early stages of the Model I TRS-80 (referred to by some as the "dark ages"), there were no such things a disk drives and expansion interfaces. All of us out in Tandyland were using tape drives (audio units, no less) with our Model Is. This meant, of course, that there was no such thing as random access files.

Because of the manner in which tape recorders worked, playing from one end of the tape to the other, data was stored and retrieved in a sequential manner. This meant that sequential file I/O was all that was allowed. And that was fine, since the sequential I/O techniques were simple and easy to master.

Let's take an example. Assume that you are a programmer writing a payroll. You decide what information you are going to keep on each person and allocate space accordingly. The simplest way to keep all the records straight is to use memory arrays. Each variable that contains important data would be arrayed such that the data for employee number 1 would be in the first element of all these various arrays.

When it was time to manipulate the data, you simply addressed the array element that corresponded with the number of the employee you were working with. The only actual "data file I/O" was done at the beginning and end of each session.

At the beginning of each session, you would input the data from the cassette sequentially and one at a time load ALL elements of the data array from the tape. At the end of the session, you would reverse the procedure. You would write the data sequentially to the cassette.

Needless to say, the main disadvantage to this method of data storage and retrieval is that it is VERY slow! It also is severly limiting in that it requires you to have enough memory overhead (memory that is not actively used for programs and hence is available to contain data) and also requires that you limit your capacity to ONLY as many as you have memory to hold.

This was not so bad with the tape machines, because tapes wouldn't hold all that much information anyway, but when the disk drives came out, it was a different story. The disk drives had two distinct advantages over tape that spelled the end of sequential data file handling for large data bases :

(1) They could hold a great deal of information as compared to the tape units.

(2) They could access data randomly. Which is to say that they did not require you to access the data in a sequential manner, but you could proceed directly to ANY piece of data that the disk was used to store.

What this meant to those of us who were programming at that time was that we had to do some significant re-vamping of our thoughts regarding data file management.

The first item that became apparent is that while we could now store enormous amounts of data for each record and retrieve that record in but a second, it wouldn't do us much good if we couldn't access the data in some orderly fashion. This requires an index.

Indexing techniques were not needed with the previous methods of programming, in which the entire data file was loaded into memory before each session. All of your data was at hand, simply sort it BEFORE you wrote it back to the tape.

There were those that tried to get by simply converting the sequential tape I/O to sequential disk I/O, but this was short-lived. With the previous technique, you were limited by memory size. Obviously, if there was no place to PUT all the data, then there was an immediate ceiling on the amount of data allowed.

With the tape drives for data storage and retrieval, there was no great concern. They couldn't hold all that much data and they were so slow with the data that they DID hold that you could not have tolerated the time needed to manipulate LARGE amounts.

The disk drives changed this. Now you have a customer who just has paid four to five hundred dollars for a little box that hooks to their expansion interface and gives them increased storage and GREATLY increased data retrieval speed. This customer would like to see the fruits of their money. To tell them that the capacity is not that much increased over tape programs and that the speed of operation has not increased as much as they think it will, does not sell many programs.

So, all of the programmers began learning about how to use random file programming and how to maximize data storage and retrieval. Initially, the techniques were similar to those used with sequential file programming.

The data would be stored on the disk in whatever format the programmer happended to arrange it, usually an aribitrary one. The program would access the data for searches and sorts in a sequential manner. For example, if you had converted that payroll mentioned earlier to this technique, you might decide that the employee number could be a disk record number rather than an array element.

Then, when you had to search for an employee by name, you would be forced to read one at a time through all of these records until you located the one that matched your search field. This, needless to say, was somewhat slower than what would be considered "optimum".

This brought up the subject of indexing again. What if all the names were stored in a file that had just the names? Then you could search that file and locate the name, access the corresponding data record, and the search time is cut drastically.

This was true to a point. The most common method was the sequential index file pointing to the random data file (can't throw those old techniques away!). You would load this index at the beginning of each session. When the search field was entered, the array would be searched instead of the disk. When a match was found, the disk record corresponding to the array element would be the one to access. Everybody KNEW that a sequential search of a memory array was faster than a sequential search of a disk file, so this seemed the way to go.

But, like many other techniques, as soon as it was developed, it was improved upon. The first thing that they did was to use a two dimensioned array that held both the key field AND a pointer to which data record was indicated. The advantage? It allowed you to SORT the index array. Then, to get a listing of your data sorted by the key field, you simply started with the first array element and continued retrieving the indicated data record until you reached the last array element. The element number itself no longer had anything to do with the position of the data record.

This was good, but NOT good enough. As the drives came with higher and higher capacity, memory began to fill up again (same old song...). Programmers found that not only couldn't they fit the DATA in memory and leave room for a program, but they couldn't even fit the INDEX and have room.

Then came the beginning of disk ISAM. Someone decided to move this array to the disk and work with it from there. Treat this index file in exactly the same manner as the memory array, but keep it on disk. Well, the "exactly the same" part didn't work so good. There DID have to be changes.

First, sorting was no longer simple. You couldn't load the file into memory to sort it, so it had to be sorted at the time of entry. This meant that you needed to search up the position of the key field in relation to the existing data (e.g. does "Adams" come before or after "Zibell"?). Once you knew where in the index this new entry belonged, you would simply "bump down" the index and make room for the insertion. The data record itself could always be appended to the end of the data file because the pointer was in the index. Nothing about the position of the data indicated the order anymore.

So, where is the problem here? Simple. Two places. First, what sort of disk search do you use to locate the postion of the data? If it is NOT a binary search, it can be improved. Notice that I did not say binary TREE search. I am referring to a simple "Hi/Lo" method of searching. Second, the bump down time was getting too long.

Why? Again simple. The key field was being stored in the index file. If that was a customer's last name, you might be talking about 20 or 30 characters. Multiply that times bumping 1000 records and you will see that you are reading and writing a GREAT deal of data. And there is no need.

Think about it. The index is ALWAYS in sorted order. The first index record is the first data record. If all you did was store the record number, you could still get your sorted listing without a problem. Simply read each index record and get the corresponding data record. However, the search was a problem.

Before, with the original search the index Hi/Lo method, you would read the middle record of the index file and check the key field against the search field and determine whether or not it comes before or after this. Then, depending on what you have found, you either restrict yourself to the lower or upper half of the file, divide THAT in half and start again. This goes on until you have located the postion.

To modify this was easy. All that was needed was to change the search routine such that when it read the index record, it simply read the data record being pointed to also and checked the search field against the key field WITHIN THE DATA RECORD. That was the key to the whole new technique. The time factor of the extra read was minimal, after all, that is what the random file structure was DESIGNED for. And the speed increase you received, compared between bumping 2 byte records or 32 byte records, more than made up for the small loss in search time.

Now that we have covered the history of these routines, we are brought up to this latest technique of file handling. Yes, that is :

The way we do things now -

Perhaps the easiest way to cover this area is to show you a sample of what it is I am talking about. Therefore, I have put together a program designed specifically to demonstrate the new technique. The following is a listing of that program :

```
10 ' ISAM style indexing technique demo program
20 ' For the DOSPLUS NEWS INFORMATION CENTER
30 ' Programmer : Mark R. Lautenschlager
40 ' Created : 11/08/82
50 '
60 CLS : CLEAR 1000
70 DEFSTR S : DEFINT I : S=STRING$(20,32)
80 PRINT "Indexing file program 1.0"
90 PRINT "DOSPLUS NEWS INFORMATION CENTER"
100 '
110 ' open and field files
120 '
130 OPEN"R",1,"Test/Inx:1",2
140 OPEN"R",2,"Test/Dat:1",20
150 FIELD 1, 2 AS D$
160 GET 1,1 : EF%=CVI(D$) ' get eof from dummy slot
170 FIELD 2, 20 AS D1$
180 '
190 ' get key field
200 '
210 INPUT @ 256,"Name : ",20,"$";SI
220 IF SI=CHR$(13) OR SI=CHR$(31) THEN CLOSE : END
230 LSET S=SI : IH%=EF%+1 : GOSUB 480 ' go look for it!
240 '
250 ' bump index file
260 '
270 IF FF%=1 THEN PRINT : GOTO 380
280 IF FF%=0 THEN PRINT : PRINT "Not found!  Bumping..."
290 FOR I=EF%+1 TO IM% STEP -1 ' set it up...
300 GET 1,I : PUT 1,I+1 ' move the record...
310 NEXT I ' until done
320 EF%=EF%+1 ' increment EOF
330 LSET D$=MKI$(EF%) ' this is where the name will go
340 LSET D1$=SI ' put name in buffer
350 PUT 1,IM% : PUT 2,EF% ' store pointer and data
360 LSET D$=MKI$(EF%) : PUT 1,1 ' store new EOF
370 GOTO 390 ' skip the FOUND message
380 PRINT "Found!  Index slot";IM%-1;" - Data slot";CVI(D$)
390 PRINT "Press ENTER to continue..."; : LINE INPUT EC$
400 PRINT @ 256,CHR$(31); ' cls
410 GOTO 210 ' loop back
```

```
420 '
430 '          Subroutine to do Hi/Lo search
440 '          Inputs S or S$ (search variable)
450 '          Inputs ih% (list length)
460 '          Returns im% (index pointer)
470 '
480 IL%=2 : FF%=0
490 IM%=(IL%+IH%)/2 ' get middle record
500 IF IH%=1 THEN IM%=2 : RETURN ' first in list?
510 IF IL%>IH% THEN IF S>D1$ THEN IM%=IM%+1 : RETURN ELSE RETURN
520 GET 1, IM% ' read index record
530 GET 2, CVI(D$) ' and get the data
540 IF S>D1$ THEN IL%=IM%+1 : GOTO 490 ' higher?
550 IF S<D1$ THEN IH%=IH%-1 : GOTO 490 ' lower?
560 FF%=1 : RETURN ' here already!
```

Let's examine the program :

Lines 60-90

These lines contain the opening information and definitions. I usually define "S" as indicating a string variable and "I" as being integer, but you are welcome to do otherwise. The actual explicit variable "S" is defined in line 70 as a string of 8 spaces. We will use this later when setting up for the search.

Lines 130-170

These lines contain the code that opens and fields the data files. Please note that although we are working with very small files here for the sake of simplicity and speed, the data file may expand to hold as much information as you need.

In line 160, we are retrieving the end of file counter from the first index record. Using random files, we must always be aware of just how many records are in these files are valid. Random files never get any smaller (unless you use Todd's "Shrink/Bas"), so this must be stored by the program.

The way that I handle it here is to use the first record of the index file as a "dummy" record. That is, the data it contains is not directly related to the position of any data record. We will compensate for this one record offset later in both the search and add routines.

Lines 210-230

These lines input the data to be stored from the screen. I use the Input@ function here. If you recall last issue's article on Input@, you know that Input@ can return either of two control characters (as opposed to string data). In line 220, I am checking for these. If you press either ENTER or CLEAR at the prompt, I will close the data files and end the program.

Line 230 contains the setup for the search routine. The first item is to LSET the input variable into the search string. The reason for this is that as we read records from the data file for comparison, these variables will be padded with blanks (a standard function for a file buffer variable). In order for this to compare correctly with the variable input from the screen, we must pad that with blanks, also. The easiest way to do this, although not always the best, is to use BASIC's LSET function and let it do all the work.

The second thing that we do is to set the length of the index file to be searched. Since this file can contain more actual records than are valid, due to deletions and such, we maintain this manually. Our search subroutine gets this in the variable "IH%". The variable "EF%" was obtained from the dummy index record earlier. We set this to "EF%+1" because of the fact that the index IS one record offset due to the dummy record.

Lines 270-410

These lines contain the screen display and add record routines. In line 270, if "FF%" is equal to "1", then the input string already exists and it skips to a routine that informs you of this. Line 280 informs you if this is not the case.

Lines 290-310 contain the actual index bump routine. Again, note that we use "EF%+1" for the length of the index in order to compensate for the first record being used as a dummy. The variable "IM%" is returned from the search routine. This variable points to one of two things. Either the record that points to the data record that the information is in, or if the data is not found, it contains which record in the index should have the information. We are bumping in reverse order, so the "STEP -1" is there to see that the values decrease instead of the normal increase.

In line 300, we read the record pointed to by "I" and then write it into the next higher record. This begins with the last record and continues until the record where the new item belongs has been reached and moved to make room.

In lines 320-360, we have the actual data storage routine. Line 320 is incrementing the EOF. Never do this until you are ready to write the records. Accidentally incrementing that EOF variable can cause the sorts, searches, and bumps to all go awry. In line 330, we set the index pointer to reflect where the data will be stored. We are going to store the data in the record pointed to by "EF%". Since the data itself is not sorted at all, but rather written sequentially, the next available data record is always the number of valid records plus one. Since we just incremented that variable, we can use it to position the data.

Line 340 stores the data in the permanent buffer. Line 350 puts the index record into the slot that we bumped open for it and then appends the data record to the end of that file. In line 360, we store the updated EOF variable back in the first record of the index.

Line 370 simply skips around the "found" message, because if program control gets to line 370, then the information was NOT found. Line 380 prints a message if the data is found to already be in the file. It tells you both what index slot it is found in and what data record that slot points to. Please note that I subtract 1 from the index slot in order to compensate for that index being offset by one.

Lines 390-410 simply print a message on the screen requiring the user to press ENTER to continue and then clears the input area and loops back for another.

Lines 480-560

These contain our Hi/Lo search routine. Line 480 is the setup line. In it, we set the variable "IL%" to 2. That points to the first record in the list to be searched. If you decide to store your EOF variable somewhere other than the first record, simply set this to 1. Remember to remove all that other offsetting throughout the program if you do. This line also resets the found flag, "FF%".

Line 490 determines the record between the high point "IH%" and the low point "IL%". This is regarded as the middle point "IM%". Line 500 checks to see if the index is empty. If the EOF was 0, then our offset would set this to 1. At that point, there is no need to go through the search. It is going to go in the first slot (#2) anyway. So, we just set the pointer to slot 2 and return.

In line 510, we check to see if the low point is greater than the high point. This will occur when we have run out of index before finding the record. If the record is NOT found, we will always return from here. The check to see if the search variable is greater than the file variable is to poisition correctly in the index.

If a record is not found, when we realize this we could be positioned at the record above the position to insert at or at the record below. We always want to point to the upper boundary. Therefore, we check to see if we are on the low boundary (i.e. is the search variable greater than the last file variable?). If this is true, then we increment the pointer to reflect the upper boundary and return. If we are already positioned on the upper boundary, we simply return.

Line 520 reads the index record and line 530 reads the data record that the index points to. Line 540 checks to see if the variable read from the data file is greater than the search variable. If it is, then the low point is moved up to the previous middle point. Line 550 checks for just the opposite. It determines whether or not the search variable is less than the file variable and then, if this is true, decrements the high point by one. Both of these two routines pass control back to line 490 where the new middle point for these adjusted boundaries is calculated.

If the file variable is neither greater or less than the search variable, then it is equal to it. If they are the same, then the search item already exists in the file. Line 560 then sets the found flag to 1 and returns.

So you see, the technique for indexing is really very simple. It can be outlined as follows :

> Maintain an index that contains only a pointer to the data.
>
> Always have this index sorted in whichever order you wish to preserve your list.
>
> Keep the EOF information in variables. Don't trust it to the LOF function of BASIC. We are dealing with random files here, and as we insert and delete information, the actual EOF value will change even if the file length doesn't.
>
> Use a Hi/Lo style search to locate the data (or the position to insert the new data) in the file. If the entry is not found, bump the index to make room for the record, place it there, and append the data record to the end of the data file.

Deleting from these files -

Oh yeah, I almost forgot. You probably are curious as to what is the recommended method of deleting from this sort of file structure. Well, it is as simple as adding the information in was.

The procedure is essentially this : First, you search up the record to be deleted in the index. Store that slot number somewhere. Get the last record in the data file and move it into the position that the record to be deleted now occupies. This piece of data is contained in the index record that you just read. If the record to be deleted just happens to be also the last record in the file, all you must do is decrement the EOF.

Then, reverse the bump procedure that we had above. Instead of bumping from the current record into the NEXT record, bump from the current record into the PREVIOUS record. You are bumping the slot closed, not open.

When the bumping of the index is done, you have one of two options. First if the record deleted was the last record in the file and all you did was decrement the EOF, then you are finished. Close the files and return. If, on the other hand, you DID move a record into another record, you must search up the index pointer for the record that you moved and adjust it to point to that record's new location.

In either case, you will decrement the EOF pointer by one.

That's all there is to it. This is not a hard technique to implement. If you use this style of BASIC programming, you should be able to garner sufficient speed for your BASIC subroutines that writing them in machine language is not needed. I hope that you can make use of this demonstration.

Let me close with this. Regarding this article, I cannot or will not :

> Accept phone calls.
>
> Look at programs that you have written using these techniques to see what it is that you have done wrong and correct it for you.

This may seem hard-nosed to some of you, but there are simply too many of you who read the newsletter for you all to get my individual attention. If you have a question regarding these file handling techniques, you may write to me care of the DOSPLUS NEWS INFORMATION CENTER and eventually, I will answer you. But I cannot allow phone calls on the subject or get involved in debugging your software. This demo program will run and other software I have written does, too. The routines work as designed. Thank you.

- Ed.

Random Routines
By Todd N. Tolhurst

This month we present an article describing a technique to shorten random-access files, and a small machine-language program that will set DOSPLUS's internal clock using the popular "TCHRON" or "TRSWATCH" hardware clocks.

TIMESET/CMD

This program, contributed by Samuel J. Porter of Boswell Bay, MI. The program sets DOSPLUS's internal time clock from the TCHRON, or TRSWATCH hardware clock for the TRS-80. The program listing is shown in Fig. 1.

TIMESET/CMD can run on the Model I or the Model III. Type the program into a Z80 editor/assembler and assemble the program. To use the program, just type "TIMESET" from the DOSPLUS command level, and the program will be installed in high memory.

The Incredible Shrinking File

One question that DOSPLUS owners often ask is "How can I shrink a random-access disk file to recover wasted or unused disk space?" Well, as everyone knows, you can't shrink a random file; it may get bigger, but it'll never get smaller.

At least that's the conventional wisdom. This month, we'll investigate a technique of shrinking random file from a BASIC program, although the principles apply equally well to other high-level languages and to assembly language.

Before we can write a program to shrink random files, we must first understand how DOSPLUS "knows" the size of a file. Every time a file is OPENed inder DOSPLUS, the system stores a great deal of important information about the file in an area of RAM known as the Device Control Block, or DCB (sometimes called an FCB, or File Control Block). Each file OPENed under DOSPLUS has a unique DCB associated with it. For more information on DCB's and the information stored therein, refer to the DOSPLUS 3.4 Technical Manual.

In this article, we are primarily concerned with the file length data stored in the DCB. The DCB stores the file length in two parts:

```
;
;         TIMESET/CMD
;
;         WRITTEN BY: SAMUEL J. PORTER
;
PORT      EQU     112                ;TCHRON BASE PORT ADDRESS
;
SET1      EQU     4410H              ;SET INT ROUTINE (MOD 1)
SET3      EQU     4413H              ;SET INT ROUTINE (MOD 3)
TIME1     EQU     4041H              ;TIME STORAGE (MOD 1)
TIME3     EQU     4217H              ;TIME STORAGE (MOD 3)
VER       EQU     0125H              ;MOD 1/MOD 3 MACHINE VERSION
;
          ORG     5200H
;
START     LD      A,(VER)            ;GET MACHINE VERSION
          CP      'I'                ;MOD 3?
          LD      HL,TIME1           ;ASSUME MOD 1
          LD      B,2                ;MOD 1 INT SLOT
          LD      A,10               ;MOD 1 COUNT
          JR      NZ,START0          ;MOD 1
          LD      HL,TIME3
          LD      B,11               ;MOD 3 INT SLOT
          LD      A,3                ;MOD 3 COUNT
START0    LD      (TIM2+1),HL        ;SET TIME ADDRESS
          LD      (TIMUPD+2),A       ;SET INT COUNT
          LD      (TIMUPD+3),A       ;SET INT COUNT
          LD      HL,SET1            ;ASSUME MOD 1
          JR      NZ,START1          ;MOD 1
          LD      HL,SET3            ;MOD 3
START1    LD      A,2                ;INT SLOT
          LD      DE,TIMUPD          ;ROUTINE
          JP      (HL)               ;SET INT TASK
;
          ORG     0FF00H
;
TIMUPD    DEFW    TIM1               ;ROUTINE ADDRESS
          DEFB    0                  ;DELAY 1 SEC
          DEFB    0                  ;INIT COUNT VALUE
;
TIM1      DEC     (IX+2)             ;DECREMENT DELAY
          RET     NZ                 ;NOT YET
          LD      A,(IX+3)           ;GET COUNT VALUE
          LD      (IX+2),A           ;INIT DELAY
;
TIM2      LD      HL,$-$             ;HL=>SECONDS
          LD      C,PORT+1           ;C=SECONDS PORT
          CALL    BYTE               ;GET SECONDS
          INC     HL                 ;HL=>MINUTES
          CALL    BYTE               ;GET MINUTES
          INC     HL                 ;HL=>HOURS
          CALL    BYTE1              ;GET HOURS
          INC     C                  ;SKIP DAY OF WEEK
          INC     HL                 ;HL=>DAY
          INC     HL
          CALL    BYTE1              ;GET DAY
          INC     HL                 ;HL=>MONTH
          CALL    BYTE               ;GET MONTH
          DEC     HL                 ;HL=> YEAR
          DEC     HL
;
;         CONVERT BCD BYTE TO BINARY
;
BYTE      IN      A,(C)              ;A=MSB DIGIT
          JR      BYTE2              ;SKIP
;
BYTE1     IN      A,(C)              ;A=MSB DIGIT
          AND     3                  ;IGNORE
BYTE2     LD      E,A                ;E=MSB DIGIT
          ADD     A,A                ;*2
          ADD     A,A                ;*4
          ADD     A,E                ;*5
          ADD     A,A                ;*10
          LD      E,A                ;SAVE MSB DIGI
          DEC     C                  ;C=>LSB DIGIT
          IN      A,(C)              ;A=LSB DIGIT
          ADD     A,E                ;ADD MSB
          LD      (HL),A             ;SET BYTE
          INC     C                  ;C=> NEXT BYTE
          INC     C
          INC     C
          LD      A,(VER)            ;MOD 1 OR 3?
          CP      'I'
          JP      Z,3591H            ;MOD 3
          RET
          JP      3591H
;
          END     START
```

Fig. 1

(1) The total number of complete 256-byte sectors occupied by the file. This is called the ERN, or Ending Record Number. It is a two-byte value stored at DCB+12.

and

(2) The number of bytes occupied in any partial sector. This is called the EOF, or End of File byte. It is a one-byte value stored at DCB+8.

From this information, we may calculate the length of the file in bytes (let's call it "L"), using the following equation:

L = (ERN * 256) + EOF

or, conversely, we may compute the ERN ond EOF, given a file length:

ERN = INT(L/256)
EOF = L - (ERN * 256)

Now, let's say that we have a file with a LRL, or Logical Record Length, of 57. If the file were 350 records long, the length of the file in bytes would be:

57 bytes/rec * 350 recs = 19,950 bytes

Therefore, the ERN for this file would be:

INT(19,950/256) = 77

and the EOF would be:

19,950 - (77 * 256) = 238

now that we have learned how DOSPLUS stores file lengths, and how they are calculated, how do we apply this to our avowed purpose of shrinking a random file? Easy. All we need to do is decide how many logical records we would like the file to contain, calculate the actual length in bytes, convert that into an ERN and an EOF and then place the new ERN and EOF into the file DCB and CLOSE the file.

"That is easy", you say. "But wait! Where do I find the DCB?" Well, in assembly language they're easy to find, since the programmer places them wherever he wishes. BASIC, on the other hand, puts its DCB's anyplace it feels like, and worse yet, it doesn't tell us where. Ah, but we have ways of making BASIC talk . . .

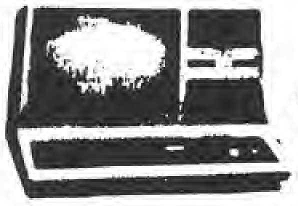
If you refer to the BASIC program in Fig. 2, lines 500-530 illustrate a method for locating a BASIC file DCB. The routine function in this way: First, we use the VARPTR function to locate the "string descriptor" for the variable S0 (the first, and in this case, the only variable in the FIELD statement for the file in question). The address returned by VARPTR points to the length of the string, followed by the actual RAM address in which the string data is stored. Since this variable exists in the disk I/O buffer itself, this address is the address of the file's I/O buffer. We can use this information to find the address of the file's DCB. How? As it happens, BASIC always creates a 32-byte file DCB followed by a 256-byte disk I/O buffer. Therefore, once we have located the start of the disk I/O buffer (variable X2) all we need do to find the DCB is subtract 32 (variable ID).

Lines 600-640 perform the calculations to determine file length, ERN and EOF, and lines 700-720 POKE these values into the file DCB. The file is then CLOSEd, and DOSPLUS automatically "shrinks" the file to your specifications.

These principles may be used in any BASIC application that might benefit from the unused space that can be reclaimed from random-access files. Happy shrinking!

Fig. 2

```
10'
20'  SHRINK/BAS
30'
100 CLEAR 1000 : DEFSTR S
110 CLS
120 PRINT"File Shrinker"
130'
200 INPUT@128,"Filespec:",23,"$";SN
210 INPUT@192,"LRL      :",3,"#";SL
220 INPUT@256,"NRECS    :",5,"#";SR
230'
300 IL=VAL(SL):'logical record length
310 IR=VAL(SR):'number of records
320'
400 OPEN"R",1,"SN",IL:'open file with proper LRL
410 FIELD 1, 1 AS S0:'field buffer
420'
500 X1=VARPTR(S0):'locate pointer to S0
510 X2=PEEK(X1+1)+PEEK(X1+2)*256:'locate S0 (buffer)
520 IF X2>32767 THEN X2=X2=65536:'adjust if >7FFFH
530 ID=X2-32:'DCB address
540'
600 L=IL*IR:'file length in bytes
610 ER=INT(L/256):'ERN
620 EO=L-ER*256:'EOF
630 E1=INT(ER/256):'get MSB of ERN
640 E2=ER-E1*256:'get LSB of ERN
650'
700 POKE ID+8,EO:'put new EOF into DCB
710 POKE ID+12,E2:'put LSB of ERN into DCB
720 POKE ID+13,E1:'put MSB of ERN in DCB
730'
800 CLOSE
810 GOTO 200
```

BASESCRIPT + YOU +  = SPEED RELIABILITY EASE OF USE FLEXIBILITY

Florida Micro Computer Systems

BASESCRIPT IS AN ENTIRELY NEW CONCEPT IN WORK PROCESSING PACKAGING. INCLUDED IN THE PROGRAM ARE THE FOLLOWING FEATURES -

Mes

- A FULL FEATURED LETTER PROCESSOR
- A COMPLETE MAILING LIST SYSTEM
- A MAIL MERGE PROGRAM TO PRODUCE MULTIPLE PERSONALIZED LETTERS
- A FORMS PROGRAM WHICH ALLOWS FAST FILL OUT & UPDATE OF USER CREATED FORMS.

SOME OF THE FEATURES OF THE LETTER PROCESSOR ARE-

- EASE OF USE (BUILT IN HELP COMMAND AVAILABLE DURING EDITING)
- AUTOMATIC WORD SPILLOVER CORRECTION
- AUTOMATIC CENTERING & MARGINS
- PAGE NUMBERING & HEADERS
- MATH FUNCTIONS WITHIN THE WORD PROCESSING DURING ON SCREEN EDITING
- MEMORY REMAINING REPORT
- CHARACTER, LINE, AND BLOCK INSERTION/DELETION
- CONTINUOUS OR PAUSE BETWEEN SHEET PRINTING
- BASIC LANGUAGE PROGRAM FRAME WITH MACHINE CODE ROUTINES WHERE SPEED IS REQUIRED (GIVES BOTH EASE OF MODIFICATION & MACHINE LANGUAGE SPEED)

THE MAILING LIST SYSTEM IS A COMPLETE & FLEXIBLE SYSTEM DESIGNED TO BE USED WITH THE MAIL MERGE PROGRAM TO PRODUCE MULTIPLE PERSONALIZED LETTERS. THE MAILING LIST SYSTEM ALSO INCLUDES A PROGRAM TO PRINT LABELS &/OR ENVELOPES. IT CAN ALSO BE USED AS A STAND ALONE MAILING LIST WHICH IS USER DEFINEABLE INTO 8 SEPARATE LISTS.

THE FORMS PRODUCTION PROGRAM TAKES FORM DOCUMENTS CREATED WITH THE WORD PROCESSING PROGRAM AND ALLOWS FAST FILL OUT, UPDATE, & PRINTING OF THE FORMS USING AN AUTOMATIC INSERTION POINT SEEK ROUTINE.

THE MAIL MERGE PROGRAM AUTOMATICALLY INSERTS DATA INTO FORM LETTERS CREATED USING THE WORD PROCESSING PROGRAM. DATA CAN BE SUPPLIED BY THE COMPUTER DIRECTLY FROM THE MAILING LIST OR MANUALLY FROM THE KEYBOARD. IN THE COMPUTER SUPPLIED MODE, DISCRIMINATION CAN BE MADE IN THE PRINTING SELECTION BY ZIP CODE &/OR ANY COMBINATION OF THE 8 USER DEFINED SUBLIST CATEGORIES.

ALL OF THE ABOVE PROGRAMS ARE CONTAINED ON 1 (ONE) 5 INCH DISKETTE AND WILL OPERATE UNDER DOS PLUS 3.4 OR 4.0 RUNNING ON MODELS I AND III AND ON LNW80 MICROCOMPUTERS. THE COMPLETE PACKAGE, INCLUDING A HARD BOUND 40 PLUS PAGE MANUAL SELLS FOR $99.95 TO EXISTING DOS PLUS OWNERS, OR FOR $229.95 INCLUING A COMPLETE DOS PLUS 3.4 OPERATING SYSTEM.

Phone 971-9300 **Florida Micro Computer Systems** 1631 West McNab Road Ft Lauderdale, Florida 33309

THE KING OF UTILITIES SUPER UTILITY PLUS

"I believe SUPER UTILITY or SUPER UTILITY PLUS should be present at every TRS-80 disk installation."

We didn't say this; Paul Wiener did in 80 Microcomputing, Jan. '82...but we sure agree with him!

You heard about it! You read about it (80 Microcomputing). Now get the "cadillac" at a special price!

Compatible with MOD I, and MOD III, and all the current operating systems! Copy files from any DOS to any DOS, MOD I or III, without converting!

Zap
Purge
Format
Special Format
Disk Repair
Memory
File Utility
Tape Copy
Format without erase
Disk Copy
Special Disk Copy
Configurable System

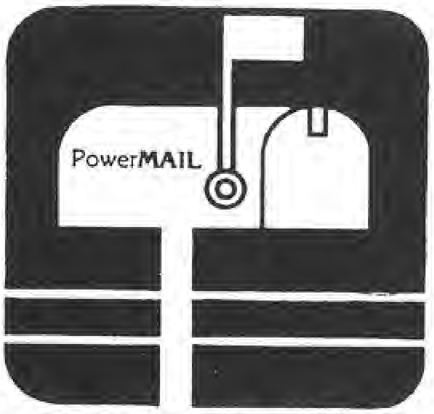
MUCH MORE   Mod I & Mod III on Same Disk

For MOD I/III . . . $74.95

NEW

Back up copy **NOW** included.

Also Available:
Super Utility Plus Tech. Manual . . . $14.95
"Inside Super Utility Plus" $19.95



THE MOST POWERFUL, FLEXIBLE DISK MAILING SYSTEM FOR THE TRS80

SUPPORTS 65,000 NAMES

PowerMAIL is a highly sophisticated mass mailing system designed to run under all of the popular DOS's currently available for the Mod I or III. The program is written entirely in machine language for maximum operation speed, and occupies only 4K of the available RAM in your computer. There are no 'slow' periods when PowerMAIL is running. New features have been added to the program that others have always lacked. You now have the ability to keep track of mailings using the 24 'flags' that are incorporated into the PowerMAIL program. The PowerMAIL system will handle a file up to 8 megabytes, or 65535 names, whichever is smaller. The program will run in as little as 32K and one disk drive, although 48K and 2 drives are desirable. The program will also sort the entire maximum file size and open up to 168 files simultaneously during the process. Author Kim Watt.

For MOD I/III...$99.95

Hundreds of Satisfied Users...Scriplus 3.0

Scriplus is a modification to Scripsit** which enables you to take advantage of the special functions, features, and print formats of your printer while your document is being printed. Allows you to:

```
change expanded print
change no. of characters per inch
or underline in mid-line:
```

Features
- Compatible with all current DOS's (I or III)
- Modifies ALL versions of SCRIPSIT**
- Allows usage of MOD I version on MOD III
- Allows MOD III versions to be BACKED UP for your protection.
- Files can be killed, loaded, merged, or chained from the Scriplus directory.
- Scriplus supplies an ALPHABETIZED directory with FREE space shown.
- "END" returns to DOS READY instead of rebooting.
- Printer can be stopped for insertion of text or forms alignment. Inserted text can be edited prior to resumption of printing.
- Specifically written for the MX-80 but will work with any printer that accepts CHR$ codes for control.
- Optionally select line feed after carriage return.

For MOD I/III . . . $39.95

This Special Limited Edition Package will be in high demand as only 500 copies will be made. They will be numbered 1-500 and will be personally signed by the author, Kim Watt. YOUR name will be embedded in the program as the serial number. The following is included with this SPECIAL LIMITED PACKAGE:

1) SUPER UTILITY PLUS S/E in /CMD File Format. Both MOD I and III versions are included, and your NAME will be the serial number. This will NOT be a protected disk, and you may make as many BACKUPS as you wish. The serial number is NOT changeable.

2) TWO attractive SU+/SE binders.
Binder #1 will include:
Three manuals in LARGE format (8 1/2 x 11")
(a) SUPER UTILITY+ Manual
(b) INSIDER SUPER UTILITY by Paul Wiener/ foreward by Kim Watt
(c) SUPER UTILITY TECH Manual by Kim Watt & Pete Carr

3) Binder #2 will include THE SOURCE CODE for SUPER UTILITY PLUS.

Yes...the SOURCE CODE to this MAJOR program will be available to 500 programmers. This is FULLY commented by the author, Kim Watt, and is a machine language programmer's dream come true! After reading this, your machine language programming skill should increase tremendously. All of Kim's knowledge in ONE book! All at your disposal and for YOUR use.*

4) The license to USE Kim Watt's sub-routines... will be granted to those 500 registered owners! These 500 ONLY will be able to apply all of Kim's magic to THEIR programs. No royalty fee necessary. In other words, IMPROVE YOUR PROGRAMS! Take Kim's ideas and expand on them! Never has anything EVER been done like this before. These 500 ONLY have the right to use our sub-routines. This information is NOT being put in the public domain. We are allowing these 500 to use our routines by buying our special package...All copy-rights and trademarks are retained by Breeze/QSD, Inc.

5) SU+/SE is NOT available from any dealer, but only directly through Breeze/QSD, Inc. Customers will be handled on a one-on-one basis. Confirmed orders will be pre-registered and a matching card must be returned by purchaser for full support from Breeze/QSD, Inc. We will know who each and every owner is, so full support can be given. We DO want you to sign and return our registration card for this support to commence however. No exceptions will be made.

6) This is a very important step that we are taking, and only a select group can appreciate the value in a package like this. This is NOT for the general mass market. It is a college education in machine language written by a recognized expert. It IS SU+ in /CMD file form. It is a license to use Kim Watt's subroutines. It is an opportunity to vastly improve your product. It is a collector's item, also. Limited, indeed. Last, but not least, it is expensive. On the surface only, however, as this product will make you an expert programmer if that is what you want. You can literally write a DOS from studying the code! It will also make you a member of an elite group that has access to Kim's knowledge and can USE that knowledge to YOUR benefit.

Source Code is FULLY Commented.

Price for the Super Utility Plus-Special Edition Is $500

Available later this year
Call or write for more information

*Credit to Kim Watt and Breeze/QSD must be given in the program and in the documentation for sub-routines used. There is NO royalty fee to pay however.

POWERSOFT
A DIVISION OF BREEZE/QSD, INC

11500 STEMMONS FWY.
SUITE 125
DALLAS, TEXAS 75229
(214) 484-2976

ORDER TOLL-FREE 1-800-527-7432

Welcome to our NEW column!

Since "Karl's Klassroom" is our newest feature column here in the DOSPLUS NEWS INFORMATION CENTER, I thought that we might take this opportunity to meet the man behind all this. Not only is this man the only REAL professional writer working with us on this staff, but he is also perhaps the most qualified person to discuss computers in education I have ever met. I've asked him to prepare a small piece to introduce himself, and he has. So, without further ado, my good friend "S. W. Karl" :

S. W. Karl is one of the many pseudonyms of a South Florida writer who has written 84 novels, numerous magazine articles, a stage play and two screen plays. With degrees in classical studies, foreign languages and history, he spent seven years teaching, then left the classroom to work as a salesman of almost anything that could be sold--securities, real estate, insurance, industrial lighting and advertising specialties. He was regional sales manager in charge of hiring and training salesmen for a national motor club when he quit and turned to a career as a full-time, professional writer. As a writer, he abandoned his typewriter (after wearing out three of the things) in favor of a word processor. In the past year he has stealthily crept back into the field of education, teaching adult education courses on How to Write A Novel, BASIC Programming, and Word Processing. He has been employed as a word processing consultant, training employees in the operation of four word processing systems for the TRS-80 Model III and Model II computers. Currently teaching programming, word processing and computer operations for a large, international corporation, he still manages to devote two hours a day to writing novels. He is sometimes assisted in his writing by a 23-pound cat named Clyde, who enjoys parking on his lap and watching the cursor run back and forth across the screen. (Clyde has been known to add editorial comments while attempting to catch the little bugger!)

S.W. admits to being a voyeur in his spare time, reading other people's mail on public BBSs. He claims this is his only vice. (Not true. -Ed.) We hope KARL'S KLASSROOM will provide a channel of communication for our owners who are in any field of education and who would like to exchange ideas for upgrading their profession.

KARL'S KLASSROOM
by S. W. Karl

It is no surprise to anyone that sales of microcomputers today are making the most enthusiastic forecasts of just a year or two ago seem conservative. TRS-80 Model III's and Model II's are a common sight in every conceivable business environment; a small business without a computer today is marked as being behind the times, and the proliferation of good software gives the businessman the edge he needs to keep costs down and efficiency at a high level.

Because the business community has been acquiring computers at such a rapid rate, the educational community has become aware of the need to train young people in the use of microcomputers before sending them out into the competitive world lacking the skills needed to be employable. Two years ago only universities, colleges and junior colleges had offerings in "computer sciences" for their students. Most of the courses taught were for the purpose of training programmers, systems analysts and engineers. Very few students were able to elect courses in applications of microcomputers such as Word Processing or Electronic Accounting. It was a rare school that had computers in classrooms to assist teachers and students in academic subjects. The few educational programs available were seldom used by schools; they were considered a luxury that could be afforded only by the rare students who had a micro at home.

The situation today has done a complete turnaround, and not only high schools but even elementary schools are being equipped with classrooms where the computer is as familiar a sight as the pencil sharpener. Pre-schoolers are learning more than they are playing while sitting in front of a micro. Most games teach coordination of eyes and hands, but children want more than simple games when they sit down at a computer. My wife and I recently had friends in to spend a leisurely Sunday afternoon with us, and their six-year old son (who has difficulty concentrating on any studies because of emotional problems) first played a few arcade-type games on my Mod III, and was bored with each of them after five to ten minutes. I then introduced him to Alphakey(tm), a Radio Shack program to teach the letters of the alphabet and the layout of the keyboard. When it was time to eat, I had to turn off the power to get him to come to the table with us. He'd been sitting in front of that thing for two and a half hours!

Because children learn so readily and are so fascinated with the machines, school systems from coast to coast are refiguring their budgets, with most of them allocating as much money to the installation of computers as they can squeeze out of federal, state and local funding. One private, church-related school in Central Florida purchased 12 Model III's and a Network 3(tm). (The Network 3(tm) is an RS232C networking system which permits all student stations having cassette-type machines to access the disk drives and printer at the teacher's console. Software for the Network 3(tm) was developed by Micro Systems Software, Inc.) Originally the school planned to use the classroom to teach Fundamentals of Microcomputers and BASIC Programming to the high school students. On Saturday after the system was installed, I was called on to conduct an educational workshop to acquaint all the teachers in the school with the computers and give them an inkling of how they could use them in the various disciplines. The enthusiasm of the teachers was nearly as great as that of the students, and the school has set up a committee to find the funds to equip several classrooms in the elementary grades with computers, and double or triple the number in the upper division.

The infusion of computers into the classrooms has created a situation that is paralleled in the business community. The average small business buys a micro, and the already overworked, suspicious secretrary/bookkeeper finds herself facing a strange (and usually unfriendly) machine; she is given a four-foot stack of manuals to study (which speak of bits, bytes, hexadecimal and video op-codes); if she is lucky, she is shown where and how to turn it on. "The salesman said it'll cut your work in half, Sophie, so don't be afraid of it. It's just like a typewriter with a couple of aces up its sleeve. I don't think you'll have any trouble with it. Let's aim at having it up and ready to do the payroll and all the correspondence by the first of the month, shall we?"

Any secretary who can maintain her composure under such circumstances deserves a gold medal! I'll never understand why a man will spend $10,000 to buy a system that will save him money, and refuse to spend $49 to train the people who will use it.

Most school teachers fare better, although their situation is far from ideal. In some school systems, training of the teaching staff in operation of the machines begins at the time the decision is made to purchase the machines, in the hope that by the time the systems are delivered the teachers will know how to use them. Unfortunately, there is an over-emphasis on having all teachers learn to program in BASIC. This makes no more sense than having a student driver learn petroleum chemistry before learning how to drive an automobile. PILOT, AUTHOR, LISP and FORTH are good languages for teachers, for they can be learned in a very short time, and the languages are designed for CAI (Computer Assisted Instruction). Show me a teacher who can't master PILOT in two hours and I'll show you a teacher who can't pass a fifth grade reading test!

In the vast majority of cases, teachers will never write their own instructional materials, just as they will not write the textbooks they use. What they need is a choice of well-prepared software that can be booted and executed. I have seen some excellent materials produced by Random House and South Western Publishing. The teachers who are able to choose from good software of this kind are fortunate. Too many have little or nothing at their disposal, and some will have nothing except what they will be able to develop on their own. Not only is this situation difficult for the teachers, it is cheating the students for whose benefit the computers were purchased.

Because of this, there is a crying demand for <u>good</u> educational software. Any programmer who has a background in the field of education stands an excellent chance of reaping the rewards that are going to be paid for developing the programs that will enable Johnny to read, write and master his arithmetic. Educational software is Big Business today, but as in any other form of publishing, the independent author who can produce something the big houses can sell is in demand. And purchasing special programs to be used by one teacher or one department is not unusual. Most teachers will purchase a program with their own money just as they will buy a reference book, if it will help them in their job of training their students.

Among the many thousands of registered owners of DOSPLUS there are teachers who have developed programs to help them in the classroom. Some of these programs are as simple (or sophisticated) as record-keeping aids for attendance and grade-averaging. They are, in reality, specialized data base management programs. I have seen one program that was written by a teacher who proudly claimed it as his first attempt to program something functional. No matter how inelegantly written, the program does what it was intended to do: it is a seating chart that enables the teacher (and his colleagues) to keep track of which student occupies which seat in any given period in any one of fifteen classrooms!

Good programs are the result of people like you who find a need and fill it with their own computers. Perhaps you've found a listing of a program in one of the popular magazines that you have adapted in some manner to suit your needs, or taken sections of several programs and merged them to perform a task for you that none of the original authors could have foreseen. I know I'm never satisfied with simply using off-the-shelf software--I like to customize it. (As good as Todd's Macro-Key program is, I tinkered with it before I was satisfied. May he never know!) I like user-friendly prompts and error messages. I get irritated with a program that tells me to HIT ENTER. Most of these lines have been changed to PRESS ENTER. Nobody hits my computer!

This column is being written with the hope that it will become a clearing house of ideas and programs by educators for educators. We at Micro Systems Software want to hear from you. How about it? We'd like to see your programs, and hear how you've solved your problems. What you have written just might be what hundreds of others need. Write to me in care of the DOSPLUS NEWS INFORMATION CENTER.

Software reviews

This next section is another new section in the DOSPLUS NEWS INFORMATION CENTER. We have not had a regular review column in here, but this issue marks the debut. Each issue will have reviews on software that is designed for or simply runs on the DOSPLUS family of operating systems.

If you have a program that you would like to see reviewed here, this is the procedure :

(1) Send in a copy of the program, complete with documentation. Send it to "Attention software reviews". A reviewer will then be assigned to the program.

(2) Provide us with information such as upgrade policies (where applicable), pricing, interface to hardware (where applicable), and potential uses. Write us a letter introducing the program. We want to be as familiar with the program as possible so that we can give it an honest evaluation.

(3) Provide us with a telephone number or address where we can reach you should problems or questions arise.

I promise you that we WILL be honest. Whether or not that is a good thing depends on the quality of the program. If you have a review already prepared for your product, you may send that also. Be certain to include permission (in writing) to publish this. If the review is not grossly out of line with what the product seems to be, we may publish that.

This month, we are reviewing three programs :

(1) Holman Large Capacity Accounts Receivable

(2) LSO - Library Support Option

(3) RIPcheck

Next month, we have a review on a new program from TSM Enterprises aimed at Amway distributors. If you want to get your program in the next issue, send it today. If it arrives too late, it will wait.

Software review

Program :
Large Capacity Accounts Receivable

Company :
Holman Data Processing Services
2059 West Lincoln
Oroville, CA 95965

Pricing and machine :
Retail price $99.95 (after 01/01/83 - $149.95)
Model I or III TRS-80 (supplied on TDOS)

Reviewer :
Mark R. Lautenschlager

Overview :

This program is written for the TRS-80 Models I and III and is supplied, ready to run, on a copy of our TDOS kernel system. The program is written entirely in BASIC with no machine language subroutines. The program files are non-protected and the media is backup-enabled.

The program is a single installation accounts receivable. Single installation means that it is a program aimed at the end user, not an accounting firm. It maintains the accounts receivable for one location; the computer it is installed upon.

It is a stand alone system, not interfacing to any other accouting software. It does not work from an inventory, but rather a list of transaction codes supplied by the user.

Specifics :

This program's claim to fame is its extremely high capacity. The author, Bill Holman, has put the ability to span disks into this program. It functions on a two disk system, but allows up to 31 data disks to be swapped in and out of drive 1. The capacity resulting from this is enormous. It will handle up to 5000 customer accounts with 15000 transactions. These are astounding figures to anybody used to the conventional accounts receivable programs.

You have 99 transction códes that be used for anything you wish. For most people, this will be enough. This program, being a stand alone accounts receivable, does NOT interface to any variety of inventory. But then, it's not supposed to do that. You will use the transaction codes to describe the various goods and services that you will be providing your customers with.

There are also 18 different customer status codes. These let you set up 18 different ways for your customers to pay and place any one of them on the proper schedule just by entering the proper code. You may use these to have the program print the proper immediate payment amount on the invoice. For example, if you had a customer who just created a $100 dollar invoice with you and you had defined that account as paying 1/3 balance each month over a period of 90 days, the invoice would reflect this. It would instruct the customer to pay $33.33 with this invoice as well as reminding them of the original $100.00 purchase.

The program is very user friendly and runs well. Please note that I actually reviewed the "test set" as opposed to the full blown system, so the maximum number of accounts I had was 30. It seems to implement fairly efficient code, though, and I can see no real speed problem as the system fills up.

The flow of data in the system was logical. You enter customer accounts and transactions. Following that, you post the transcations to the customer's account. You may then produce statements or reports regarding your customers. The program is not what you would call a visually elegant one, but the menu options are clean and functional and data entry was simple.

The program IS a menu driven one and that makes it easier for most users to operate the program on a daily basis without having to constantly refer back to the documentation.

There are quite a few program modules and these are all stored on drive 0. The master index and other pointer files are also on drive 0. The transaction and customer data is stored on the disk or disks located in drive 1.

Positive points :

There were several things that I liked about the Holman Data Services program. First off, I liked the fact that it is as "user configurable" as it is. Most of the accounts receivable programs on the market today list a set of values that they use for default levels in the various areas of the program and you are expected to like that. If you do not care for it, use someone else's program. Bill Holman allows you to configure almost every area of program/customer interface.

The high capacity is also nice. Some users, albeit few and far between, will require this many accounts and/or transactions. For them, this is perhaps the only Accounts receivable they can use.

It has an interesting feature in the "Quick customer status report". This, without generating a piece of paper, will give you the complete status of any customer account. This can be useful for those quick "spot checks" that you will need to make from time to time.

The system seems fairly well debugged. I experienced no problems in any of my testing that I could attribute to program errors. It will verify each data disk switch and will not proceed unless the correct disk is in the drive.

The sub account feature is nice. You can have major accounts and then link in these divisions of that account. The number appended to the standard account number with a decimal point is all that it takes.

This may seem like a minor point to some, but the program WILL function in a 32K TRS-80. Many business programs rely on full blown machines. If you do not have all the RAM you can and you don't want to spend the extra money, this could be a factor.

The system handles credit entries well and displays them with the conventional notation (i.e. "<>"). When entering a credit, you may place the minus sign before OR after the number (i.e. -123.45 or 123.45-). It is little items like this that mark the user friendliness that was programmed into this.

The reports are fairly complete and well laid out. All reports will print on 80 column paper. This is a departure most welcome from the standard "You have to use a 132 column printer!" approach that many software packages (even, alas, some of our own) seem to take.

Negative points :

That's right, negative points. No program is perfect and this is no exception. It does have some shortcomings.

The most serious shortcoming may or may not affect how you feel about the program. I spoke with the author, Bill Holman, on the telephone and he informed me that although he has written BOTH an Accounts Receivable and a General Ledger, they do not interface and he has no intention of making them do so. He also has no intention of supporting this system with any other software. I find this very disappointing. I like to know that there will be not only ongoing support for the program I buy now, but also future software that I can buy to enhance my package.

Bill even stated that at this time he has no plans for a version that does not use the "disk splitting" method. This means that the program will not function in a hard disk environment. It seems like it would be a simple thing to alter the program so that it wouldn't split the data files every so often and allow them to expand on the multi-megabyte hard disks.

This program also has an annoying habit of printing something out and then not executing a top-of-form. And there is no convenient key to press to cause it to happen. It is true that some people do not like having the program top-of-form all the time, feeling that it is wasting paper. But if you want a page advance, there should be an easy way to achieve it. I usually have my programs top-of-form when you press the <up arrow> key. Having a Daisy Wheel II can make this especially aggravating, because there is no button to push. You have to advance the form manually EVERY time!

The documentation was also a lowlight. It bills itself as being an easy to understand tutorial manual. I found it to be neither. I thought it was confusing and poorly packaged (26 pages of manual, 14 pages of print samples, all done in dot matrix and stapled together). I would rate the documentation and packaging sub-standard to say the least.

It is a VERY good thing that the program runs as easily as it does, or else there would be a lot more problems arising from it than there are. However, with this program, the manual is almost superfluous. But let me say this. In this day of "enlightenment" regarding the quality of documentation, packaging, and support, there is NO excuse for a manual that is not at least run on a daisy wheel printer. The appearance of the manual can set the tone for the entire operation of the program.

Summary and conclusion :

This program is a good one. All things considered, it does what it is supposed to, and what it is ADVERTISED to do, all with a minimum of operator effort (except for the $%&'#" paper!). That is itself is a rarity for the user to find.

If you are looking for an integrated business package, look elsewhere. If you are looking for fancy documentation, look elsewhere. If you are looking for future support software, again, look elsewhere.

But if you are looking for a simple balance forward accounts receivable with a user-friendly, error-trapped style about it that will handle enormous amounts of data in a relatively rapid manner, then this just could be the program that you want.

P.S. - Bill Holman has made a special offer regarding his program to registered DOSPLUS owners. For a limited time, he will send you the TEST SET which has retailed nationally for $30 to $50 for only $10. This is an actual subset of the program, so for $10 you will see what you are getting. It does include all the features of the regular program and the manual. In addition, he is offering a 15% discount off the retail price of the full blown package. Just tell him that you saw it here.

Software Review

Program :
LSO - Library Support Option

Company :
XYZT Computer Dimensions Inc.
2 Penn Plaza
Suite 1500
New York, NY 10121
(212) 244-3100

Pricing and machines :
Retail price $79.95
Model I or III TRS-80 (all DOSes)

Reviewer :
Linden Avery

Overview :

This program is distributed on a formatted data diskette compatible with all DOS systems. The file is non-protected and the media is backup-enabled. The program is of the type that you will install it on all system disks that you intend to use it with.

The LSO program is the latest offering from XYZT Computer Dimensions in New York. Their previous venture into the TRS-80 marketplace was ICL - Interactive Control Language. For those of you who are great fans of DOSPLUS' DO command, you should see what sort of things you can do with ICL.

LSO is best described as being a "partitioned data set" program. It allows you to "pack" up to 255 small files into one larger file, using only one actual directory entry. These files are dynamically allocated and deallocated by the LSO program, such that the "packing" and "unpacking" are invisible. With LSO in as a background task, you never need know that all this internal file manipulation is going on.

Specifics :

LSO is self-relocating and sits in high memory. It intercepts ALL I/O requests except for RENAME. For those of you who are technical "buffs", that is because RENAME is NOT a vectored call. There is no way that LSO can intercept it. To avoid problems, they supply you with LNAME/CMD, part of the LSO "utilities". This program replaces RENAME. What all this intercepting means is that all of your standard DOS library commands work with the "packed" files just as normal.

If LSO intercepts an I/O request and doesn't locate it in one of their "libraries", as XYZT has dubbed the larger files, it will pass the request back to the DOS and things will proceed in their normal course. For example, assume that you are packing all of your BASIC programs (extension /BAS) into the library "BAS/LIB". LSO is resident and you are currently in BASIC. You give BASIC the command LOAD"TEST/BAS". First, this command is intercepted by LSO and the library "BAS/LIB" is searched. But the file is not found there. At that point, LSO sends the request back to the DOS and DOSPLUS looks for a file called "TEST/BAS" on the disk (unpacked) and loads it in as normal.

Effectively, this means that you have the best of both worlds. You can use packed and unpacked files on the same disk. Because LSO is completely automatic, NO additional expertise is required to operate it. No further command syntax is required.

LSO also includes a utility called "LSET" that allows you to turn on and off certain options about each library. The first option is the "permanent open". This option leaves the library file open after accessing it. What this effectively means is that there is some small speed increase, since LSO doesn't have to open its library before commencing I/O to it. XYZT warns that you may NOT remove a disk with a permanently open library on it, because the next time that you access that library, it will not check the directory for the location of the library but rather will proceed to its last known location. This option does have some limited usefulness.

The other option is the "compressed format". When LSO is storing data into a library, if it finds three or more characters the same, it will compress those into a single byte indicating the type and number of characters to follow. This seems of highly limited usefulness. Even in text files, it is VERY rare that you will have too many sequences of characters the same that the applications program does not compress itself. It may be useful in space compression function.

<u>Positive points :</u>

LSO has a lot going for it right off the bat. It is very automatic (for the most part, I wasn't even certain that is was running). As a matter of fact, LSO is SO transparent that XYZT has to flash an "&" sign in the upper right hand corner of the screen during disk I/O to let you know that it is there.

LSO functions "as is" with the DOSPLUS 4.0 hard disk systems! This is a bonus beyond belief for hard disk users. Many of us, myself included, have 5 or 10 meg hard disks running under DOSPLUS 4.0. We have found that it is easier to go through 255 files than 5 meg. With LSO, this problem is solved for good.

I can have up to 255 LSO libraries in a directory, each one of those with 255 files in it. For those of you without calculators, that is 65025 files per logical volume. That ought to hold us for a while, right? And again, because LSO is so automatic, the only additional steps are (1) creating the libraries with LCREATE and (2) loading LSO before a session.

Creating an LSO library is simple. All you must do is enter LCREATE ***/LIB:*, where the "***" is the three letter extension that all files in this library will have in common and ":*" is the drive on which to place it. As you have now surmised, the method used by LSO to determine which files are in which library is the filename's extension. And you may NOT have two LSO libraries with the same name. Obviously, the DOS will not allow it, since two libraries for the same extension would have the same exact filespec. Not to mention that if LSO doesn't find the file in the library, it sends the I/O request back to the DOS without further ado.

Although you can only have 255 files with any one extension, nothing is preventing you from storing them under extensions like "D01" for the first 255 data files and "D02" for the next, etc., etc. A little creativity with the extensions will go a long way here.

Does everything sound just a little TOO good? Sort of "Alice in Wonderland time"? Well, no program is perfect.

Negative points :

LSO's documentation. The program is SO automatic, it almost doesn't need a manual at all. And the one that they have is sufficient. That is, IF you can overlook the writing style. The manual is written as if the author had but a marginal grasp of the English language and grammatical structure.

The information is there and it IS clear, but the writing style proved often amusing and, hence, detracting.

There is one rather bizarre side effect of using LSO. LSO seeks to open all files for read and load them. Therefore, you cannot have any files with a higher protection level than read status. That means no "execute only" files. This includes the DOSPLUS utility files. Even though Micro-Systems has given us the password (For those of you who missed it, it is "XANTH" - Ed.) it is still a large inconvenience to have to use the ATTRIB command to lower the protection level on all those files.

The size of your individual libraries is only 255 sectors maximum. This can seem somewhat limiting to hard disk users. However, there are two points to consider. First, the large, constantly changing data files are what the hard disk was designed to handle. These files can efficiently exist in the standard DOS environment. Second, the only files that you would wish to "pack" will be smaller files that tend to be far more numerous. For those, LSO works fine.

There is also this ray of hope. After being given a review copy of the program by Micro-Systems, I called their technical department to ask what was being done. I was told that they had spoken with XYZT about it and a new version was planned shortly that would allow up to 65000 records in a file. I am looking forward to this as, at that point, this program will then please ALL of the people ALL of the time.

Summary and conclusion :

LSO is a fine piece of software worth its weight in gold. The modest retail price is more than fair for such a useful utility. If you have an application wherein "packing" files could be useful, this is the best that I have seen.

The documentation WAS a bit bizarre, but it was clear and contained all the information one could want.

LSO is a good investment, especially for the hard disk user or the user with large capacity floppy drives. If you have EVER filled up a directory before a disk, this is the answer.

Software review

Program :
RIPcheck Check Register and Accounts Payable program

Company :
Glenn/Cliff Associates
8301 East Montebello
Scottsdale, AZ 85253
(602) 941-0609

Pricing and machine :
Retail price <not given>
Model III TRS-80

Reviewer :
Sharon Wampler

The creative programmers at Glenn/Cliff Associates have come up with an excellent check register/writer/payables system for the TRS-80 Mod III/I, designed around the super DOSPLUS 3.4 operating system. RIPCHECK is self-prompting and simple enough in operation to delight the computer neophyte who just wants a checking account system, while at the same time it has the capacity and capability to handle the needs of a small business as a checkbook, check-writer and payables system. It is extremely fast in operation, and can handle up to 4 separate checking accounts, each account having up to 39 vendors (payees) and up to 560 records per year.

RIPCHECK consists of 6 menu-driven programs which load as needed. From the initial boot-up, the program senses the account status and will automatically run every program needed to set up the user accounts. Filespec information is retained in high memory to facilitate the interaction of the various sub-programs and increase the speed of operation. It has conveniences like optional automatic check numbering and simple self-defined codes (such as "R" for Rent) to help users. Mistakes are correctable and records can be deleted, voided and modified. Vendor files can be updated and maintained from year to year. There is a special year-end program which allows you to update the vendor data for the new year.

As a checkbook system, your data is retrievable at any time. Machine-level-speed sorts and searches give you quick access to reviewing your data. You can review by month, by month and code, by code, or with a very helpful instring search for a keyword. Hardcopy of the information can be obtained by using screen prints.

RIPCHECK is a sophisticated computer-printed check-writer for those wishing that feature, or checks can be written to the screen if you prefer to hand-write your checks. A complete check register is printed after each session.

As a payables system, there is a recap of unpaid checks and the user can request a listing of those due in the next 30, 60, etc., days. Undated payables are flagged. The program also flags how long a check is past due.

RIPCHECK is visually delightful with well-designed, attractive screen displays which make it a pleasure to use. It is user- friendly with extensive built-in error traps, helpful screen notes and features like automatic backup routines. It is fast in operation and accurate to payments to $9,999.99. For more information, contact Ted at Glenn/Cliff Associates.

Questions & Answers
By Todd N. Tolhurst

Questions featured in this column are representative of the questions received by Micro-Systems Software Technical Support. Address specific inquiries to:

Micro-Systems Software, Inc.
Technical Support Division
4301-18 Oak Circle
Boca Raton, FL 33431

Q: I have two double-sided drives on my Model III, drive 0 and drive 1. When I try to create a double-sided system diskette for drive 0, I CONFIGure drive 1 for SIDES=2 and then I format a diskette in drive 1. All of this works well, but when I try to SYSGEN the diskette in drive 1, the SYSGEN program runs for a few seconds and then reports a "Directory Read Error". What am I doing wrong?

A: Actually, you're not doing anything wrong. The problem is with your double-sided drives. For some reason, when DOSPLUS instructs the drive to access the second side of the diskette, the drive is still using the first side. During a SYSGEN, DOSPLUS attempts to write system programs to the diskette, and some of these program fall on the same track number as the diskette directory, track 17 decimal, but on the opposite side of the diskette. If the drive is not responding to DOSPLUS's request for the second side of the diskette, this results in the disk directory being overwritten by SYSGEN, causing the directory read error.

This problem is often caused by using a drive cable that does not have pin 32 (side select) attached, or when using drives that have been modified by the vendor to act as if the double-sided drive is actually two separate drives. In either case, have a good technician check out your system and correct the side select problem.

Q: I use DISKZAP to examine and sometimes alter the contents of disk sectors. When I use my double-sided drives, I can't seem to get DISKZAP to display any sectors above 11H, even though I'm supposed to have 36 sectors/track on a double-sided double-density disk. Can't DISKZAP work on double-sided diskettes?

A: Of course it can. First, let's clarify one point: You do not have 36 sectors/track on a double-sided double-density diskette. You do have 18 sectors per track, just like on a single-sided double-density disk. The number of sectors per track is not affected by the number of surfaces available on a diskette. You do have 36 sectors per cylinder. A cylinder consists of all like-numbered tracks on a diskette. On a double-sided floppy diskette, two tracks comprise a cylinder.

Cylinders are artificial constructs used by the DOS. From the floppy disk controller's level, the double-sided drive has two surfaces (we'll call them A and B), each of which contains 18 sectors. DISKZAP operates on this level. DISKZAP knows nothing of cylinders, nor of files or directories, or any of the special structures maintained by the DOS. DISKZAP simply displays and edits disk sectors. In order to do this, you must tell DISKZAP which drive, surface, track, and sector you wish to display. We specify the surface by including the letter along with the drive specification (surface A is assumed if only the drivespec is entered). For instance, to display track 22, sector 8 on surface B of drive 1, answer DISKZAP's prompts like this:

```
Drive?  1B
Track?  22
Sector? 8
```

When you wish to display a sector number greater than 18 (on a double-density system), remember that the sector resides on surface B, and that the actual sector number is 18 less than the number provided by the MAP utility or other similar programs. For instance, sector 24 decimal is actually on surface B, sector 6.

Log off
Mark R. Lautenschlager
Editor-in-Chief

MicroNET Support NOW Available to DOSPLUS Users!

Are you a member of Compuserve? If you are, you may be familiar with the various SIGs (Special Interest Groups) that are available. XTRA-80 (sponsered by PowerSOFT) is the second oldest SIG that MNET has. Its membership is over 800 members and growing all the time. There are ten sections devoted to the support of various products/hardware. We are pleased to announce that there is now a section for Micro Systems Support, and that should be of interest to many of you. Todd Tolhurst, our illustrious master of all answers, will be checking on a frequent basis to electronically support DOSPLUS and get you an answer ASAP. There is also a section for MicroPOWER, which will support the Mod II/16 DOSPLUS II. Kim Watt and Micro Systems will monitor THAT section. As a little bonus, there is a data base of over 800K of programs, routines, and information for your downloading pleasure! There are 10 sub-sections in the Sig and they include:

DOSPLUS Support
SNAPPWARE Support
PowerSOFT Support
Lance Micklus, Inc. Support
MicroPOWER Support

These sections are manned by the people actually involved in the companies represented. In most cases you will get a personal answer from either Lance Micklus, Bob Snapp, or Mark Lautenschlager and Todd Tolhurst from Micro Systems Software. The PowerSOFT section is manned by Dennis Brent, Renato Reyes PhD, and is monitored by Kim Watt. We also have hardware sections devoted to:

Mod I/III
Mod II - Mod 16
IBM PC

Other experts in the field frequent our board and it is not too often that a question goes un-answered whether it is hardware or software related! Check it out! From the "OK" prompt, type,"R QSD <enter>". We are here to support you! See you there! In order to be logged in as a full fledged member, you will need to send Micro Systems Software your MicroNET PPN#. We will confirm it against our registered data base (only registered owners get on free), and we will forward these on to PowerSOFT. - Ed.

New personnel at MSS

Micro-Systems Software Inc. would like to take this opportunity to welcome two more talented people to our "team".

Richard Leun, our new customer service representative, has undergone extensive training in computers before coming to bring complete support service to all Micro-Systems Software owners. He is available at (305) 983-3390 and also accepts correspondence at our business office.

Due to the ever increasing Micro-Systems Dealer Network throughout the world, we have appointed Christine R. Walczak as our new Dealer Sales Coordinator. With her extensive background as a sales representative of a large, international computer manufacturer, she is eminently qualified to aid you in your purchasing. If you would like to join the Micro-Systems Dealer Network, please contact our sales department at (800) 327-8724.

Crystal Ball Deparment

If you have been listening to the gossip that leaves these hallowed halls (and there IS a lot of it!), you are aware that we have been waiting anxiously for one of our hard disk OEMs to develop a multiplexor and send it to us for the software. To date, none have.

We have ceased to wait. Micro-Systems has developed our OWN multiplexor. It will function on the Model III TRS-80 and will allow you to interface up to four Model IIIs to a single hard disk. A special version of DOSPLUS 4.1 will be provided with each unit.

Retail price figures have not been set at this time, but we are anticipating something higher than $700 and less than $1000. You will be able to "daisy chain" more than one unit together to interface more than four Model IIIs. It will also interface to most major brands of hard disk, not being tied to any particular brand.

These units are being manufactured for us under exclusive contract by a major electronics firm based here in South Florida, so they will be TOP quality products. We should have the first units available for the Model III sometime in January. This unit, if popular, will be adapted to other micros as well.

Contact Larry Studdard (it was HIS pet project...) here at Micro-Systems for any further details.

Next issue, another NEW column!

In this issue, you saw the premier of two new features, the software reviews and KARL'S KLASSROOM. In the next issue of the DOSPLUS NEWS INFORMATION CENTER, we will be unveiling yet another new entry.

For those of you with DOSPLUS II, we will be starting a yet unamed column on technical tips and programming with the DOSPLUS II Disk Operating System. This will be a regular feature here and will be authored by none other than Kim Watt, one of the program authors. Kim will be added to our newsletter "gang" as the resident Model II/16 editor.

DOSPLUS II technical support update

Many of you who have purchased the DOSPLUS II Disk Operating System have questions regarding it. Many of you have called the numbers in the manual and posted messages to the CompuServe PPNs listed.

Well, I goofed. The numbers listed for PowerSOFT are incorrect. The telephone number SHOULD be :

(214) 484-2976

and the CompuServe PPN should be :

70001,610

The information listed in manual is incorrect and if it is answers you wish, please use the correct number for support.

Support IS available either through Micro-Systems OR PowerSOFT. If the telephone is busy at one location, try the other. We will try to help you if at all possible. The numbers are (again) :

(305) 983-3390 - Florida
(214) 484-2976 - Texas

Connection terminated

Well, faithful reader, that is all that I have for you this issue. We have enjoyed, once again, all the madness and hectic "fun" that is involved in bring you this publication.

We are re-naming this publication for 1983. I am soliciting any and all suggestions. Everything but the "Tandy Newsletter" is applicable.

Please check the hard disk coupon that is buried in the back here. $1599 for a COMPLETE hard disk. No hidden anything! Remember, a five megabyte drive makes a GREAT stocking stuffer. I even put the coupon where it won't hurt you to cut it out. Not one precious (grin) word of this text will be lost.

Until next issue, remember : This is DOSPLUS country!!!

Mark

ACEMAIL

The Evolution of Low-Cost Electronic Communications

We believe AceMail is probably the most powerful, automated two-way message system available for anywhere near the price. But don't let the price fool you.

The Automated Computer Electronic Mail software offers some unprecedented clout in communications across the phone lines. From a completely unattended TRS-80, AceMail can look up a phone number, dial it, log/on to the system, upload and download ASCII files, log/off and wait until later to call another computer. You may also call AceMail from a remote location and exchange information.

A world of information is at your fingertips as AceMail offers speed and accountability in accessing branch office correspondence, sales reports, data banks and time sharing systems nationwide.

The multifunctional system is easy to use and comes with a complete users manual allowing even the inexperienced user to send and receive electronic mail in minutes!

If your TRS-80 is due to evolve in the world of electronic mail, then choose the communications system with ease and power—AceMail.

Written exclusively for the Hayes Stack Smartmodems and the TRS-80 Model I/III 48K with disk. AceMail operates on NEWDOS/80 or DOSPLUS and comes supplied on 'tiny' DOSPLUS ready to run.

Hayes Stack Smartmodem 1200—$619.00
Hayes Stack Smartmodem 300—$239.00
ACEMAIL 1200 Software—$119.00
ACEMAIL 300 Software—$79.00

ace computer products of florida, inc.

1640 n.w. 3rd street, deerfield beach, florida 33441 voice (305) 427-1257 data (305) 427-6300

ORDER TOLL FREE 1-800-327-2283

DOSPLUS 4.0, the perfect score.

The DOSPLUS 4.0 hard drive system is here!

A hard drive without the dynamic new DOSPLUS 4.0 is like an eggshell without the egg. The new DOSPLUS 4.0 is the leading edge—the latest in the line of advanced disk operating systems from MICRO-SYSTEMS SOFTWARE, INC. It's the only current operating system written from the ground up for hard disk operation. Not just a driver but a fully developed system. What that means to you is more bang for your buck! Finally an unlimited TRS-80 for small business. For only **$1899** you get the DOSPLUS "PLUS."

THE DOSPLUS 4.0 FEATURES

- Single volume addressing/Double sided floppies seen as one drive–one file can expand to limit of the hard drive
- Hard Disk—disk editing utilities
- Incredible I/O speed
- Runs any combination of densities or tracks
- Also operates 8" drives with special hardware—comes with expanded users guide and complete DOS technical section on I/O calls and DCB organization
- Ability to use hard drive as the "system" drive.

"PLUS" MANY OF THE SENSATIONAL NEW DOSPLUS 3.4 FEATURES

- BASIC array sort—multi key, multi array
- Tape/Disk—Disk/Tape utility (with relocator)
- Input (controlled screen input)
- Random access and ASCII modification on Diskdump
- BASIC checks for active "DO"
- Backup and Format from a "DO" file
- Much improved Backup (More reliable)

$1899 TAKE YOUR MICRO TO THE MAX. ORDER NOW!

COMPLETE WITH 5 MEG SYSTEM

- I/O package much faster (disk access time reduced)
- Repeat last DOS command with "/" [ENTER]
- Short directory (filename and extension) available
- Short directory of Model III TRSDOS disks
- Single file convert from Model III TRSDOS
- COMPLETE device routing supported (DOS and BASIC)
- Ability to save BASIC programs directly to another machines' memory (if equipped with DOSPLUS 3.4)

NOTE: The final versions of 3.4 and 4.0 will have almost identical features and documentation.

THE COMPLETE SYSTEM

- Smooth, silent, swift
- Error-free disk I/O
- *Add on up to 4, 10 meg units for a total of 40 megabytes!
- Plugs on the 50 pin data bus.—no loss of floppy drives
- Completely self-contained—just plug it in and go

*10 meg units available Soon.

NOTE: Specify 40 or 80 track when ordering DOS diskette. After initial bootup, user can create any DOS desired.

The first in the industry backed by a lifetime warranty**

**Lifetime warranty on original media

MICRO-SYSTEMS SOFTWARE, INC.
4301-18 Oak Circle
Boca Raton, FL 33431
Telephone: (305) 983-3390
800-327-8724

DATED MATERIAL
Please do not delay!

Bulk Rate
POSTAGE
PAID
Permit No. 1
Hollywood, FL